(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 578 059 A1**

(12) # EUROPEAN PATENT APPLICATION

(21) Application number: **93110069.7**

(51) Int. Cl.[5]: **H04L 9/32**

(22) Date of filing: **24.06.93**

(30) Priority: **30.06.92 EP 92401869**

(43) Date of publication of application:
12.01.94 Bulletin 94/02

(84) Designated Contracting States:
DE ES FR GB IT

(71) Applicant: **THOMSON CONSUMER ELECTRONICS S.A.**
9, Place des Vosges,
La Défense 5
F-92400 Courbevoie(FR)

(72) Inventor: **Naccache, David**
46, rue St. George
F-94700 Maisons-Alfort(FR)

(74) Representative: **Wördemann, Hermes**
Deutsche Thomson-Brandt GmbH,
Patents and Licensing,
Göttinger Chaussee 76
D-30453 Hannover (DE)

(54) **Method for executing number-theoretic cryptographic and/or error-correcting protocols.**

(57) Different methods for access control, digital signature and identification are known which use modular multiplications. A protocol is presented which can be executed in the same time as a Fiat-Shamir scheme where all modular multiplications are replaced by standard multiplications. The advantage of the new method over the classical Fiat-Shamir protocol is that computations are done in a much quicker way.

EP 0 578 059 A1

The present invention relates to a method for executing number-theoretic cryptographic and/or error-correcting protocols.

Background

Montgomery's algorithm described in "Modular Multiplication without Trial Division", Mathematics of Computation, vol 44, pp 519-521, is a process for computing $A*B*2^{-|n|}$ modulo n in O(Log(n)) memory space (O: order).
In Fiat, Feige and Shamir, "Zero-Knowledge Proofs of Identity", Journal of Cryptology, vol 1, pp. 77-94, an authentication scheme (Fiat-Shamir) is described.
See also EP-A-0252499 and EP-A-0325238.

Invention

It is one object of the invention to disclose a method of constructing a Fiat-Shamir-like authentication scheme suitable for Montgomery environments without introducing any overhead in the number of modular multiplications requested for the execution of the normal protocol. This object is reached by the method disclosed in claim 1.

A very recent result described in Arazi, "Modular Multiplication is Equivalent in Complexity to a Standard Multiplication", Fortress U&T Internal Report (1992), Fortress U&T Information Safeguards, P.O. Box 1350, Beer-Sheva, IL-84110, Israel, establishes (in a constructive way) that $A*B*2^{-|n|}$ mod n can be computed with the same complexity (timewise and hardwarewise) as A*B (not mod n).
This theoretical reduction of the problem of modular multiplication, recently applied to the design of today's fastest hardware modular multiplier, is very important since it implies that the protocol presented hereafter can be executed in the same time as a Fiat-Shamir scheme where all modular multiplications are replaced by standard multiplications.
The fact that no constants are to be precalculated beforehand and the small amount of RAM requested for software implementation of the new protocol makes it highly convenient for smart-card applications, e.g. in pay TV systems.
The advantage of the new method over the classical Fiat-Shamir protocol is that computations are done in a much quicker way. Also, the same philosophy can be applied in order to transform or adapt other number-theoretic schemes for quick execution.

All along this application, $|n|$ denotes the length of n (in bits) and $\|n\|$ the Hamming weight of n.
In Montgomery's algorithm for modular multiplication mentioned above it is assumed that n is odd. No other restrictions are imposed on the modulus.
Very much simplified, this algorithm 'Kernel' works as follows:
Let X[i] denote X's i^{th} bit (with X[0] as LSB) and $K = 4^{|n|} \bmod n$.

```
Kernel(A, B)
{
c = 0
For i = 0 to |n|-1
   If A[i] == 1 then c = c + B
   If c[i] == 1 then c = c + n
   c = c/2
If c ≥ n then c = c - n
Return(c)
}
```

It can be shown from the Montgomery reference cited above that $c = A*B*2^{-|n|} \bmod n$ and that:

Kernel(K, Kernel(A, B)) = A*B mod n.

A more comprehensive and complete approach is presented by Arazi mentioned above, where it is formally shown how to reduce the complexity of the computation Kernel(A, B) into that of A*B.

If it is possible to transform the Fiat-Shamir scheme in such a way that the Montgomery parasites ($2^{-|n|}$) will not disturb the protocol (using only one Kernel operation instead of each modular multiplication), then it would be possible to perform the Fiat-Shamir scheme in about the half of the time requested with a full Montgomery multiplier.

Moreover, the precalculation or usage of the constant K of the Montgomery method is not required.

This approach to the problem is somewhat new since by opposition to the classical process of designing computational tools for comfortable execution of number-theoretic protocols, here a cryptographic scheme is transformed in order to meet a given computational limitation.

Advantageously after a proper modification of the relation between the public and the secret keys the Montgomery parasites can be used helpful instead of being disturbing.

From now on D will denote the parasite factor $2^{-|n|}$ mod n and it is assumed the availability of a half Montgomery multiplier (that is a Kernel procedure performing only the operation A*B*D mod n).

The Fiat-Shamir public v_j-s is redefined by: $D^3 * v_j * s_j^2 \equiv 1 \bmod n$.

It is assumed that the v_j's are already known to the verifier (for instance by Fiat and Shamir's F(ID,j) method).

Step 1:

A prover device picks a random number R and computes $Z = R^2 D \bmod n = \text{Kernel}(R, R)$ and sends it to a verifier device.

Step 2:

The verifier device sends the random binary vector e to the prover device.

Step 3:

The prover device computes and sends to the verifier device:

$$y = r \prod_{e_i=1} s_i D^{||e||} \bmod n.$$

This value is easily computed by:

$y = \text{Kernel}(s_{i1}, \text{Kernel}(s_{i2}, \ldots \text{Kernel}(s_{i||e||}, r))\ldots)$.

Here the ij's denote the ||e|| indices selected by vector e.

Step 4:

The verifier device computes (in a similar way to that of the previous step):

$$A = \text{Kernel}(v_{i1}, \text{Kernel}(v_{i2}, \ldots \text{Kernel}(v_{i||e||}, \text{Kernel}(y, y)))\ldots)$$

$$= y^2 D \prod_{e_i=1} v_i D^{||e||-1} D \bmod n$$

$$= r^2 \prod_{e_i=1} v_i s_i^2 D^{2||e||} D^{||e||} D \bmod n$$

$$= r^2 \prod_{e_i=1} v_j s_j^2 D^{3||e||} D \bmod n = r^2 D \bmod n$$

and tests if A = Z.

The prover device can be a smart card comprising a microprocessor and RAM means and storing in non-volatile memory means the secret keys s_i and the modulus n, whereby the smart card is connected via a smart card reader to a pay TV decoder comprising the verifier device which evaluates the public key v and stores also the modulus n in memory means.

In principle the inventive method consists in computing number-theoretic cryptographic and/or error-correcting protocols using a prover device storing in memory means secret keys s_i and a modulus n, and a verifier device which evaluates a set of public keys v_i, whereby the relation between the public keys v_j and the secret keys s_j is modified in such a way that parasites induced by the usage of a Montgomery-like modular multiplication method are met or cancelled.

Advantageous additional embodiments of the inventive method are resulting from the respective dependent claims.

Preferred embodiments

Similarly, the invention can be applied to the Fiat-Shamir digital signature method as well.

In Quisquater and Guillou, "A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory", Proceedings of Eurocrypt'88 Lecture Notes in Computer Science, Springer-Verlag (Ed. C.C. Günther) 330 (1988), pp. 123-128, a second very popular authentication scheme is described. The original Quisquater-Guillou identification scheme is the following:
The basic relationship between the secret key B and a user's (prover device) identity ID is: $f(ID)*B^v = 1 \bmod n$. Another short name for f(ID) is J. The parameter v is decided by the authority.

Step 1:

The prover device picks a random number r and sends his ID and $T = r^v \bmod n$ to the verifier device.

Step 2:

The verifier device picks a random $d<v$, computes $J = f(ID)$ and sends d to the prover device.

Step 3:

The prover device computes and sends $U = r*B^d \bmod n$ to the verifier device.

Step 4:

The verifier device checks that $T = J^dU^v \bmod n$.

This identity should hold since:

$$J^dU^v = J^d(r*B^d)^v = (J*B^{v)d}r^v = 1^dr^v = r^v = T \bmod n.$$

By applying the inventive method here the computation can be accelerated using a half Montgomery multiplier (processor). Again, the relationship between the public and the secret keys is modified: $f(ID)*B^vD^{v+1} = 1 \bmod n$, but exactly the same protocol is executed while replacing the multiplications by Kernel operations:

Step 1:

The prover device picks a random number r and sends his ID and $T = r^vD^{v-1} \bmod n$ to the verifier device. The factor D^{v-1} is added by the classical square-and-multiply exponentiation algorithm where all multiplications are replaced by Kernel procedures. More precisely, this Kernel_Exponentiation algorithm is:

```
Kernel_Exponentiation(A, p)
{
Accuml = A
x = 1
While (p not equal 0)
      {
      Accuml = Kernel(Accuml, Accuml)
      If p[0] == 1 then x = Kernel(x, Accuml)
      ShiftToTheRight(p)
      }
Return(x)
}
```

Therefrom it can be easily proved that:

Kernel_Exponentiation(A, p) = A^pD^{p-1} mod n (T is obtained by Kernel_Exponentiation(r, v)).

Step 2: The verifier device picks a random d<v, computes J = f(ID) and sends d to the prover device.
Step 3:
The prover device computes and sends $U = r^*B^dD^d$ mod n to the verifier device.
U is computed by: Kernel(Kernel_Exponentiation(B, d), r)
Step 4:
The verifier device checks that $T = J^dU^vD^{d+v-1}$ mod n.
The value $J^dU^vD^{d+v-1}$ is computed by:
Kernel(Kernel_Exponentiation(J, d), Kernel_Exponentiation(U, v))
This test should be true since:

$$J^dU^vD^{d+v-1} = J^d(r^*B^dD^d)^vD^{d+v-1} = (J^*B^v)^dr^vD^{d^*v+d+v-1} = D^{-d(v+1)}r^vD^{d(v+1)}D^{v-1} = r^vD^{v-1} = T.$$

Schnorr, "Efficient Identification and Signatures for Smart Cards", Proceedings of Eurocrypt'89 Lecture Notes in computer Science, Springer-Verlag (Ed. G. Brassard) 435 (1990), pp. 239-252, (see also EP-A-0384475) is a system where the relation between the public (v) and the secret (s) keys is $v = \alpha^{-s}$ mod p. α is chosen by the authority in such a way that $\alpha^q = 1$ mod p. q is a public key published by the authority.
Step 1:
The prover device picks a random r, computes $x = \alpha^r$ mod p and sends x to the verifier device.
Step 2:
The verifier device picks a random e and sends it to the prover device.
Step 3:
The prover device sends back y = r + s*e mod q.
Step 4:
The verifier device tests that $x = \alpha^y y^e$ mod p.
The inventive method can be applied again in order to shorten the computation time. The only modification to introduce is in the relationship between the public and secret keys which becomes:

$v^*D^{s+1}\alpha^s = 1$ mod p.

Step 1:
The prover device picks a random r, computes $x = \alpha^r D^{r-1} \bmod p$ and sends x to the verifier device.

x = Kernel__Exponentiation(α, r)

Step 2:
The verifier device picks a random e and sends it to the prover device.
Step 3:
The prover device sends back $y = r + s*e \bmod q$
Step 4:
The verifier device checks that $x = \alpha^y v^e D^{y+e-1} \bmod p$ by testing that:

x = Kernel (Kernel__Exponentiation(α, y), Kernel__Exponentia tion(v, e)).

This test should be true since:

$$\alpha^{y}v^{e}D^{y+e-1} = \alpha^{r+s*e}v^{e}D^{y+e-1} = \alpha^{r+s*e}(\alpha^{-s}D^{-(1+s)})^{e}D^{y+e-1} = \alpha^{r}D^{-(s+1)e}D^{r+s*e+e-1} = \alpha^{r}D^{r-1} = x.$$

The signature scheme disclosed in El Gamal, "A public-key cryptosystem and a signature scheme based on the discrete logarithm", IEEE Transactions on Information Theory, vol. 31, No. 4, pp. 469-472, 1985, is based on the discrete logarithm problem.
The public key is defined as an integer $p = g^s \bmod q$, where s is the user's secret key and p, q and g are public. For signing a message m, the user picks a random number k, computes $u = g^k \bmod q$ and computes v such that $m = s*u + k*v \bmod (q-1)$. {u, v} is the signature of message m.
Upon reception, the validity of the signature can be checked since

$$p^{u}u^{v} = g^{s*u}g^{k*v} = g^{s*u}g^{m-s*u} = g^{m} \bmod q$$ and m and g

are available.
In order to take advantage of the inventive Kernel__Exponentiation procedure without modifying the protocol, the public key is just redefined as $p = g^s D^{s-1} \bmod q$.
p can be easily computed by Kernel__Exponentiation(g, s). For signing, the user will pick a random k and compute $u = g^k D^{k-1} \bmod q$ = Kernel__Exponentiation(g, k), whilst the definition of v remains the same (namely: $m = s*u + k*v \bmod (q-1)$). For checking the validity of the signature, the receiver will compute:

$$\text{Kernel(Kernel\_Exponentiation(p, u), Kernel\_Exponentiation(u, v)) =}$$
$$(p^{u}D^{u-1})(u^{v}D^{v-1})D = (p^{u}D^{u-1})(u^{v}D^{v}) = (p^{u}D^{u-1})(g^{k}D^{k-1})^{v}D^{v} = (p^{u}D^{u-1})(g^{k}D^{k})^{v} = (p^{u}D^{u-1})(g*D)^{k*v} = (p^{u}D^{u-1})(g*D)^{m-s*u} = (g^{s}D^{s-1})^{u}D^{u-1}g^{m-s*u}D^{m-s*u} = g^{s*u}D^{s*u}D^{-u}D^{u-1}g^{m-s*u}D^{m-s*u} = g^{m}D^{m-1} \bmod q,$$

and will naturally compare this value to the result of:

Kernel__Exponentiation(g, m).

The Digital Signature Standard (DSS) is an upraising new standard for message signature. Very much simplified, the DSS works as follows:

Two primes, p and q, and an integer g are selected. $y = g^x \bmod p$ is computed and p, q, g and y are published. x is the user's private key.

The procedure for signing the message M is the following:

The signer device hashes message M into a compressed string m.

The signer device computes $r = (g^k \bmod p) \bmod q$ and $s = (m + x*r)/k \bmod q$ and sends the signature {r, s} to the verifier device.

The verifier device will control the signature by checking that

$$((g^{m*sE(-1) \bmod q} y^{r*sE(-1) \bmod q}) \bmod p) \bmod q = r,$$

with $sE(-1) = s^{-1}$.

In order to adapt this process to the inventive method, only the relation between the keys is modified, but two different Kernel procedures will be distinguished:

$Kernel_q(A, B)$ computing $A*B*\vartheta \bmod q$ and

$Kernel_p(A, B)$ computing $A*B*D \bmod p$.

Here is the new protocol: Redefine $y = g^{\vartheta x} D^{\vartheta x - 1} \bmod p$. The signer device computes:

$r1 = Kernel\_Exponentiation_p(g, k) = g^k D^{k-1} \bmod p$,

$r = Kernel_q(r1, 1) = (g^k D^{k-1} \bmod p)\vartheta \bmod q$ and

$s = Kernel_q(1/k, (m + Kernel_q(x, r))) =$

$\vartheta(m + x*r*\vartheta)/k \bmod q$

and sends the signature {r, s} to the verifier device.

The verifier device will control the signature by computing:

$w = s^{-1} \bmod q$

$u1 = Kernel_q(m, w) = m\ w\ \vartheta \bmod q$

$u2 = Kernel_q(r, w) = r\ w\ \vartheta \bmod q$

$v1 = Kernel\_Exponentiation_p(g, u1) = g^{u1} D^{u1-1} \bmod p$

$v2 = Kernel\_Exponentiation_p(y, u2) = y^{u2} D^{u2-1} \bmod p$

$v3 = Kernel_p(v1, v2) = g^{u1} D^{u1-1} y^{u2} D^{u2-1} D \bmod p = g^{u1} y^{u2} D^{u1+u2-1} \bmod p = g^{u1} (g^{\vartheta x} D^{\vartheta x - 1})^{u2} D^{u1+u2-1} \bmod p = g^{mw\vartheta} g^{\vartheta x r w \vartheta} D^{\vartheta x (u2+u1-1)} \bmod p = g^{w(m\vartheta + xr\vartheta\vartheta)} D^{\vartheta x (u2+u1-1)} \bmod p = g^k D^{\vartheta x (u2+u1-1)} \bmod p = D^{\vartheta x r w \vartheta + mw\vartheta - 1} g^k \bmod p = g^k D^{k-1} \bmod p$

$v = Kernel_q(v3, 1) = (g^k D^{k-1} \bmod p)\vartheta \bmod q$

and will compare that: $v = r$.

For r the signer can also compute

$r = (g^k D^{k-1} \bmod p) C*\vartheta \bmod q$

with checking by the verifier device that $v3*\vartheta*C = r \bmod q$, where C is an arbitrary constant.

For r the signer can also compute

$r = (g^k D^{k-1} \bmod p)*f(M)*\vartheta \bmod q$

with checking by the verifier device that $v3*\vartheta*f(M) = r \bmod q$, where f(M) is a public function of the message to sign.

Advantageously with no extra cost (compared to the original DSS) the security of the proposed scheme can be improved since the steps:

$r = Kernel_q(r1, 1)$ (when computing the signature) and

$v = Kernel_q(v3, 1)$ (when verifying the signature) can be respectively replaced by:

$r = Kernel_q(r1, f(m))$ (when computing the signature) and

$v = Kernel_q(v3, f(m))$ (when verifying the signature),

where f denotes any public function (e.g. |q| LBS bits of m).

With a 68HC05 type microprocessor running at 3.5MHz the Kernel operation (for 512 bit numbers) was implemented in less than 135ms, RAM usage is less than 70 bytes. A special Kernel-Squaring version runs at 85ms but requires a double RAM space.

In the article of Arazi cited above it is shown that "It is possible to compute A*B*D mod n in $|n|+1$ clock cycles. That is, a modular multiplication is performed with the same complexity (timewise and hardwarewise) as that of a standard multiplication operation".
It is thus possible to execute a Fiat-Shamir identity check (and signature, with similar modifications of the scheme) in hardware and time equivalent to that required for the execution of the protocols without modular reductions.

The same strategy of modifying the relationship between public and secret keys in order to meet or cancel the effect of parasite constants introduced by modular reduction tools can be applied to a big variety of number-theoretic authentication and signature protocols.

The invention can also be used for banking, access control or military applications.

Claims

1. Method for executing number-theoretic cryptographic and/or error-correcting protocols using a prover device - e.g. a smart card - storing in memory means secret key and a modulus, and a verifier device which evaluates public keys, **characterized in** that the relation between the public keys (v_j) and the secret keys (s_j) is modified in such a way that parasites induced by the usage of a Montgomery-like modular multiplication method are met or cancelled.

2. Method according to claim 1, **characterized in** that in the Fiat-Shamir protocol for digital signature and/or identification:
   - replacing all the modular multiplication operations by a kernel operation taking as input a couple of numbers A and B and outputting the number A*B*D mod n, where D is a constant parasite factor and n is a standard Fiat-Shamir modulus;
   - replacing the relationship between the public and the secret keys by $D^3 v_j s_j^2 \equiv 1 \bmod n$.

3. Method according to claim 1, **characterized in** that in the Quisquater-Guillou protocol for digital signature and/or identification:
   - replacing all the modular multiplication operations by a kernel operation taking as input a couple of numbers A and B and outputting the number A*B*D mod n, where D is a constant parasite factor and n is a Quisquater-Guillou modulus;
   - replacing the relationship between the public and the secret keys by $f(ID)\,(B*D)^v D = 1 \bmod n$.

4. Method according to claim 1, **characterized in** that in the Schnorr protocol for digital signature and/or identification:
   - replacing all the modular multiplication operations by a kernel operation taking as input a couple of numbers A and B and outputting the number A*B*D mod n, where D is a constant parasite factor and p is the standard Schnorr prime modulus;
   - replacing the relationship between the public and the secret keys by $v*D^{s+1}\alpha^s = 1 \bmod p$.

5. Method according to claim 1, **characterized in** that in the El-Gamal protocol for digital signature and/or identification:
   - replacing all the modular multiplication operations by a kernel operation taking as input a couple of numbers A and B and outputting the number A*B*D mod n, where D is a constant parasite factor and q is the standard El-Gamal prime modulus;
   - replacing the relationship between the public and the secret keys by $g^s D^{s-1} = 1 \bmod q$;
   - replacing the definition of u by $u = g^k D^{k-1} \bmod q$.

6. Method according to claim 1, **characterized in** that in the Digital Signature Standard:
   a) Hashing by the prover device a message M to sign into a compressed string m;
   b) Computing by the prover device $r = (g^k D^{k-1} \bmod p) \bmod q$;
   c) Computing by the prover device $s = \vartheta(m + x*r*\vartheta)/k \bmod q$;
   d) Sending from the prover device to the verifier device the signature {r ,s};
   e) Computing by the verifier device $w = s^{-1} \bmod q$;
   f) Computing by the verifier device $u1 = m\, w\, \vartheta \bmod q$;
   g) Computing by the verifier device $u2 = r\, w\, \vartheta \bmod q$;
   h) Computing by the verifier device $v1 = g^{u1} D^{u1-1} \bmod p$;
   i) Computing by the verifier device $v2 = y^{u2} D^{u2-1} \bmod p$;

h) Computing by the verifier device $v3 = v1^{*}v2 + D \bmod p$;
j) Checking by the verifier device that $v3 = r \bmod q$.

7. Method according to claim 6, **characterized in** that steps b) and j) are respectively replaced by:
b) Computing by the prover device $r = (g^{k}D^{k-1} \bmod p)C^{*}\vartheta \bmod q$;
j) Checking by the verifier device that $v3^{*}\vartheta^{*}C = r \bmod q$, where C is an arbitrary constant.

8. Method according to claim 6, **characterized in** that steps b) and j) are respectively replaced by:
b) Computing by the prover device $r = (g^{k}D^{k-1} \bmod p)^{*}f(M)^{*}\vartheta \bmod q$;
j) Checking by the verifier device that $v3^{*}\vartheta^{*}f(M) = r \bmod q$, where f(M) is a public function of the message to sign.

9. Method according to any of claims 1 to 8, **characterized in** that the operations $x = y^{power}D^{power-1}$ modulo N are done by an exponentiation procedure, the algorithmic structure of which is:

```
Accuml = y
x = 1
While (power not equal 0)
        {
        Accuml = Kernel(Accuml, Accuml)
        If power[0] == 1 then x = Kernel(x, Accuml)
        ShiftToTheRight(power)
        }
Return(x)
```

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 93 11 0069

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
A	COMPUTERS & SECURITY. INTERNATIONAL JOURNAL DEVOTED TO THE STUDY OF TECHNICAL AND FINANCIAL ASPECTS OF COMPUTER SECURITY vol. 10, no. 3, May 1991, AMSTERDAM NL pages 263 - 267 D. LAURICHESSE ET AL. 'OPTIMIZED IMPLEMENTATION OF RSA CRYPTOSYSTEM' * page 263, right column, line 12 - line 23 * -----	1	H04L9/32

TECHNICAL FIELDS SEARCHED (Int. Cl.5)

H04L
G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	07 OCTOBER 1993	HOLPER G.E.E.

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P0401)